Enclosure B

Microsoft Flight Simulator 2000 Software Development Kit

Aircraft Container System

March 2000

Aircraft Container SDK Overview

The Aircraft Container system organizes Flight Simulator 2000 aircraft files and attributes so that most aircraft-related files are located close together. This logical and consistent organization makes the files easy to customize.

This document includes the following sections:

- **File hierarchy**
  Provides an overview of the aircraft container system's organizational structure.
- **The aircraft.cfg file**
  Explains the structure of and parameters in an aircraft's aircraft.cfg file, including the use of configuration sets that enable components to be shared among several aircraft.
- **The kneeboard content files**
  Explains the structure of the files that populate the tabs of the kneeboard, and how to customize them.
- **The panel.cfg file**
  Provides a brief description of an aircraft's panel.cfg file.
- **The model.cfg file**
  Explains the structure of and parameters in an aircraft's model.cfg file
- **The sound.cfg file**
  Explains the structure of and parameters in an aircraft's sound.cfg file
- **The Texture folder**
  Describes the contents of an aircraft's Texture folder.
- **Using Aliasing**
  Explains how to use aliasing to avoid file duplication by sharing components among several aircraft.

Important Notes:

- This SDK documents the aircraft container system and related files as implemented in Flight Simulator 2000. Not all of the features and functionality described herein are backwards compatible with Flight Simulator 98 and Combat Flight Simulator aircraft.
- This document contains information on how to modify and share components among existing aircraft. It does not explain how to create new aircraft.
- The .cfg files referenced in this document are simple text files ("configuration" files with a .cfg extension) and can be viewed and edited using any text editor, such as Microsoft Notepad. They can be found in the appropriate aircraft subfolders of the FS2000\Aircraft folder. You are encouraged to view actual files as you read this document, to become more familiar with the structure and syntax. *These files should only be modified--with caution--by experienced developers, as changes could render aircraft inoperable. The information included in this SDK is not supported by Microsoft Product Support.*

File hierarchy

The on-disk structure of the files in the aircraft container system organizes aircraft-related files and folders in a hierarchical and consistent way, and mirrors the way the files interact with one another. An aircraft "container" is simply an aircraft folder within the FS2000\Aircraft folder. Each aircraft folder (e.g., B737_400, Lear45, C182) contains:

- **aircraft.cfg** Specifies which model, panel, sound, texture, and kneeboard components to use, as well as certain aircraft-specific parameters.
- ***aircraftname*.air** Contains the aircraft's flight model.
- ***aircraftname*_check.txt** Provides aircraft-specific text for the Checklists tab of the kneeboard.
- ***aircraftname*_notes.txt** Provides aircraft-specific text for the Notes tab of the kneeboard, that can also be added or edited on the kneeboard itself while Flight Simulator is running.
- ***aircraftname*_ref.txt** Provides aircraft-specific text for the Reference tab of the kneeboard.
- **Model folder** Contains the model.cfg file that specifies the visual models to render during normal flight and during a crash, as well as the visual model color scheme. Also contains the .mdl visual model files themselves.
- **Panel folder** Contains the panel.cfg file that defines the layout of an aircraft's instrument panels and view parameters. Also contains associated .bmp files. (See the Panels SDK for detailed information about panels.)
- **Sound folder** Contains the sound.cfg file that describes the aircraft's sounds and the initial, minimum, and maximum volume associated with each sound. Also contains associated .wav files.
- **Texture folder** Contains textures (.bmp files) used in the presentation of the aircraft's visual model.

Note that through aliasing, more than one aircraft can access and use the same panel, model, and sound .cfg files (and hence the same panel, model, and sound component files) *without* copying the files to all the aircraft containers. For more information, see *Using Aliasing* later in this document.

The aircraft.cfg file

Introduction

The aircraft.cfg file represents the highest level of organization within an aircraft container. Each aircraft has its own aircraft.cfg file located in its container (aircraft folder). For example, the Cessna 182S aircraft.cfg can be found at

drive:\..\FS2000\Aircraft\C182\aircraft.cfg

The aircraft.cfg file specifies the versions of the aircraft included in the aircraft container, as well as what the attributes (name, color, sound, panels, gauges, and so on) are for each aircraft and where to find the files that define those attributes.

Within the aircraft.cfg file there are a number of sections. Brackets enclosing the section name identify the various sections. In order for Flight Simulator to make proper use of any variable, it is important that the variable be located in the correct section. While exact spelling is important, none of the terms are case sensitive.

To see the effects of a change, the aircraft must be reloaded. To force a reload, select a different aircraft in the Select Aircraft dialog box, and then reselect the original aircraft. (When you see the graphic of the aircraft appear in the Select Aircraft dialog box preview window, the aircraft has been selected; you don't need to click the OK button unless you want to fly.)

Errors made in creating or editing the aircraft.cfg file will show up, along with the following error messages, while an aircraft is being loaded. The error messages are listed in order; that is, the first error message represents an error early in the aircraft-loading process.

Error Message	Description
Aircraft initialization failure.	Indicates that some essential files are missing from the aircraft container. If the files are missing, the aircraft wouldn't usually be displayed in the Select Aircraft dialog box; as a result, this error is rare.
Failed to start up the flight model.	The .air file was not loaded successfully.
This is not a Flight Simulator 2000 aircraft model.	The visual model (.mdl) file for this aircraft is not compatible with Flight Simulator 2000.
Visual model could not be displayed.	An error occurred while loading the visual model (.mdl) file.

The following sections detail each bracketed aircraft.cfg section and its associated parameters. *As you read, you should look at actual aircraft.cfg files to become more familiar with the structure and syntax of them. They can be found in the individual aircraft containers, and viewed using any text editor.*

Configuration sets and their [fltsim.] sections

Each `[fltsim. ]` section of an aircraft.cfg file represents a different version (configuration) of the aircraft, and is known as a *configuration set*. If there is only one section (labeled `[fltsim.0]`), it is because there is only one configuration set in that aircraft container. If there is more than one configuration set (labeled `[fltsim.0]`, `[fltsim.1]`, `[fltsim.2]`, and so on), each one refers to a different version of the aircraft. Configuration sets allow a single aircraft container to represent several aircraft, and allow those aircraft to share components.

For instance, in the Professional version of Flight Simulator 2000, there are three versions of the Cessna 182, all housed in the same C182 aircraft container (folder). Each version has its own `[fltsim. ]` section. The182R RG parameters are listed in the `[fltsim.0]` section. The 182S parameters are listed in the `[fltsim.1]` section. And the 182S IFR parameters are listed in the `[fltsim.2]` section.

While these three configuration sets share many components, they each use different panels. The panel lines in the respective `[fltsim. ]` sections thus refer to the respective panel folder for each aircraft: panel.rg, panel, and panel.ifr. The `panel=` parameter in the `[fltsim.0]` section is `panel=rg`, the panel parameter in the `[fltsim.1]` section is `panel=` , and the panel parameter in the `[fltsim.2]` section is `panel=ifr`.

When creating and referencing multiple model, panel, sound, and texture directories, use the naming convention *foldername.extension*, where the extension is a unique identifier for that configuration set (e.g., .rg or .ifr). To refer to the folder from the relevant parameter in the aircraft.cfg file, just specify *extension* (e.g., `panel=rg` or `panel=ifr`). If a parameter isn't explicitly set (as in the `[fltsim.1]` section in the example above), it automatically refers to the default (extension-less) folder.

The parameters in each configuration set can refer to the same files, to different files, or to a mix of files. While using different panels, all three Cessna configurations use the same sounds, and thus the sound parameters in all three `[fltsim. ]` sections point to the single "sound" folder in the C182 folder.

Each aircraft defined by a configuration set will appear as a separate listing in the Select Aircraft dialog box. The fact that multiple "aircraft" share some components is hidden from the user. From a user's perspective, they are distinct aircraft (just as if all the common files were duplicated and included in three distinct aircraft containers). From a developer's perspective, the aircraft are really just different configuration sets of the same aircraft. Because they share some files, they make much more efficient use of disk space.

Within each `[fltsim. ]` section are parameters that define the details of that particular configuration set:

Parameter	Example (from Lear45 aircraft.cfg)	Description
title	`title=Learjet 45`	The title of the aircraft that will appear in the Select Aircraft dialog.

sim	`sim=Lear45`	Specifies which .air (flight model) file (located in the aircraft folder) to use.
model	`model=`	Specifies which model folder to reference.
panel	`panel=`	Specifies which panel folder to reference.
sound	`sound=`	Specifies which sound folder to reference.
texture	`texture=`	Specifies which texture folder to reference.
kb_checklists	`kb_checklists=Lear45_check`	Specifies which _check.txt file (located in the aircraft folder) to use on the Checklists tab of the kneeboard.
kb_reference	`kb_reference=Lear45_ref`	Specifies which _ref.txt file (located in the aircraft folder) to use on the Reference tab of the kneeboard.
atc_type	`atc_type=Lear`	Not used in Flight Simulator 2000.
atc_id_enable	`atc_id_enable=1`	Determines whether the user can edit the aircraft tail number from the Select Aircraft dialog. When set to 0, the tail number will not be editable from the dialog. When set to 1, the tail number will be editable. Not applicable to imported Flight Simulator 98 or Combat Flight Simulator aircraft.
atc_id	`atc_id=N700MS`	The tail number displayed on the exterior of the aircraft. This parameter can also be edited from the Select Aircraft dialog (if the atc_id_enable parameter is set to 1). Not applicable to imported Flight Simulator 98 or Combat Flight Simulator aircraft.
editable	`editable=0`	Determines whether the aircraft can be edited using the Flight Simulator 2000 Aircraft Editor application (included with Professional version). When set to 0, the aircraft will be read-only in the editor and a copy will have to be made for editing. When set to 1, the aircraft will be editable.

[forcefeedback] section

As detailed in the tables below, the parameters in the `[forcefeedback]` section of an aircraft.cfg file define the forces generated by that aircraft in the user's Force Feedback Joystick, so that different aircraft can have different force effects (a different "feel" in the stick).

Note: The values and usage of the force feedback parameters are strongly tied to the ForceFeedBack feature of the DirectInput API (DirectX, version 7.0). It's strongly recommended that you refer to the DirectX, version 7.0 SDK to help you understand the use and effects of many of these parameters. You can obtain information about DirectX version 7.0 from http://msdn.microsoft.com/directx.

Stick shaker parameters

These parameters define the simulated stick shaker force felt in the stick or yoke when flying an aircraft equipped with a stick shaker (such as the Learjet 45).

Parameter	Example (from Lear45 aircraft.cfg)	Data type	Range
stick_shaker_magnitude	stick_shaker_magnitude=5000	Integer	0–10000
stick_shaker_direction	stick_shaker_direction=0	Integer	0–35999 degrees
stick_shaker_period	stick_shaker_period=111111	Integer	0–2^32-1 microseconds

Gear bump parameters

These parameters define the simulated forces transferred from the airframe and gear drag to the stick or yoke when the aircraft's nose and main landing gear is raised or lowered (cycled). In fixed-gear aircraft this effect won't be felt because, by definition, the landing gear doesn't move. Different aircraft have different gear geometries that result in each of the gear mechanisms starting and ending its cycle at a different time. The timing deltas are brief, typically less than a second between the time that each gear starts and ends its cycle.

Parameter	Example (from Lear45 aircraft.cfg)	Data type	Range
gear_bump_nose_magnitude	gear_bump_nose_magnitude=3000	Integer	0–10000
gear_bump_nose_direction	gear_bump_nose_direction=18000	Integer	0–35999 degrees
gear_bump_nose_duration	gear_bump_nose_duration=250000	Integer	0–2^32-1 microseconds
gear_bump_left_magnitude	gear_bump_left_magnitude=2700	Integer	0–10000
gear_bump_left_direction	gear_bump_left_direction=9000	Integer	0–35999 degrees
gear_bump_left_duration	gear_bump_left_duration=250000	Integer	0–2^32-1 microseconds
gear_bump_right_magnitude	gear_bump_right_magnitude=2700	Integer	0–10000
gear_bump_right_direction	gear_bump_right_direction=27000	Integer	0–35999 degrees
gear_bump_right_duration	gear_bump_right_duration=250000	Integer	0–2^32-1 microseconds

Ground bumps parameters

These parameters collectively define a composite force that simulates the forces felt through an aircraft's ground steering controls as the aircraft travels over an uneven surface. The parameters are divided into two subgroups (numbered 1 and 2), and define the behavior of two distinct forces. The combination of the two forces (1 and 2), define a composite force that is transferred to the stick or yoke. The two forces are both sinusoidal periodic forces, with frequencies determined by the following linear equation:

frequency = (`ground_bumps_slope` * `aircraft_ground_speed`) + `ground_bumps_intercept`

The `ground_bumps_magnitude` parameters set the magnitude of the force. The `ground_bumps_angle` parameters set the direction from which the force is felt.

Parameter	Example (from Lear45 aircraft.cfg)	Data type	Range
ground_bumps_magnitude1	ground_bumps_magnitude1=1300	Integer	0–10000
ground_bumps_angle1	ground_bumps_angle1=08900	Integer	0–35999 degrees
ground_bumps_intercept1	ground_bumps_intercept1=3.0	Floating-point	0–1000000.0 cycles per second
ground_bumps_slope1	ground_bumps_slope1=0.20	Floating-point	0–1000000.0 cycles per second
ground_bumps_magnitude2	ground_bumps_magnitude2=200	Integer	0–10000
ground_bumps_angle2	ground_bumps_angle2=09100	Integer	0–35999 degrees
ground_bumps_intercept2	ground_bumps_intercept2=1.075	Floating-point	0–1000000.0 cycles per second
ground_bumps_slope2	ground_bumps_slope2=0.035	Floating-point	0–1000000.0 cycles per second

Crash parameters

These parameters define the simulated forces felt in the stick or yoke when the aircraft crashes. The parameters are divided into two subgroups (numbered 1 and 2), and define the behavior of two distinct crash-induced forces. The first force is a constant force that lasts for 0.5 seconds. After 0.5 seconds, it stops and the second force starts. The second force is a periodic square wave force; its amplitude declines linearly to 0.

Parameter	Example (from Lear45 aircraft.cfg)	Description	Data type	Range
crash_magnitude1	crash_magnitude1=10000	Sets the magnitude of the first force.	Integer	0–10000
crash_direction1	crash_direction1=01000	Sets the direction from which first force is felt.	Integer	0–35999 degrees
crash_magnitude2	crash_magnitude2=10000	Sets the initial magnitude of the second force.	Integer	0–10000
crash_direction2	crash_direction2=9000	Sets the direction from which the second force is felt.	Integer	0–35999 degrees
crash_period2	crash_period2=75000	Determines the frequency (frequency = 1/period) of the second crash force.	Integer	0–2^32-1 microseconds
crash_duration2	crash_duration2=2500000	Sets the amount of time that the second crash force is felt.	Integer	0–2^32-1 microseconds

[flight_tuning] section

Flight control effectiveness parameters

The following parameters are multipliers on the default "power" of the control surfaces. For example, a value of 1.1 increases the effectiveness by 10%. Likewise, a value of 0.9 decreases the effectiveness by 10%. A negative number reverses the normal effect of the control. Omission of a parameter will result in Flight Simulator defaulting to a value of 1.0.

Note: `elevator_effectiveness`, `aileron_effectiveness`, and `rudder_effectiveness` are adjustable using the Flight Simulator 2000 Aircraft Editor Application (included with Professional version).

Parameter	Example (from C182 aircraft.cfg)	Aerodynamic term
elevator_effectiveness	`elevator_effectiveness=1.0`	Cmde
aileron_effectiveness	`aileron_effectiveness=1.0`	Clda
rudder_effectiveness	`rudder_effectiveness=1.0`	Cndr
elevator_trim_effectiveness	`elevator_trim_effectiveness=1.0`	Cmdetr
aileron_trim_effectiveness	`aileron_trim_effectiveness=1.0`	Cldatr
rudder_trim_effectiveness	`rudder_trim_effectiveness=1.0`	Cndrtr

Stability parameters

The following parameters are multipliers on the default stability (damping effect) about the corresponding axis of the airplane. For example, a value of 1.1 increases the damping by 10%. Likewise, a value of 0.9 decreases the damping by 10%. A negative number results in an unstable characteristic about the axis. A positive damping effect is simply a moment in the direction opposite of the rotational velocity. Omission of a parameter will result in Flight Simulator defaulting to a value of 1.0.

Note: These three parameters are adjustable using the Flight Simulator 2000 Aircraft Editor Application.

Parameter	Example (from C182 aircraft.cfg)	Aerodynamic term
pitch_stability	`pitch_stability=1.0`	Cmq
roll_stability	`roll_stability=1.0`	Clp
yaw_stability	`yaw_stability=1.0`	Cnr

Lift parameter

The following parameter is a multiplier on the coefficient of lift at zero angle of attack ("cruise lift" in this context refers to the lift at relatively small angles of attack, which is typical for an airplane in a cruise condition). This scaling is decreased linearly as angle of attack moves toward

the critical (stall) angle of attack, which prevents destabilizing low speed and stall characteristics at high angles of attack. Modify this value to set the angle of attack (and thus pitch) for a cruise condition. A negative value is not advised, as this will result in extremely unnatural flight characteristics. Omission of this parameter will result in Flight Simulator defaulting to a value of 1.0.

Note: This parameter is adjustable using the Flight Simulator 2000 Aircraft Editor Application.

Parameter	Example (from C182 aircraft.cfg)	Aerodynamic term for parameter
cruise_lift_scalar	`cruise_lift_scalar=1.0`	CL0

Drag parameters

Drag is the aerodynamic force that determines the aircraft speed and acceleration. There are two basic types of drag that the user can adjust here. Parasitic drag is composed of two basic elements: form drag, which results from the interference of streamlined airflow, and skin friction. Induced drag results from the production of lift. Induced drag increases as angle of attack increases.

The following parameters are multipliers on the two respective drag coefficients. For example, a value of 1.1 increases the respective drag component by 10%. A value of 0.9 decreases the drag by 10%. Negative values are not advised, as extremely unnatural flight characteristics will result. Omission of a parameter will result in Flight Simulator defaulting to a value of 1.0.

Note: These two parameters are adjustable using the Flight Simulator 2000 Aircraft Editor Application.

Parameter	Example (from C182 aircraft.cfg)	Aerodynamic term for parameter
parasite_drag_scalar	`parasite_drag_scalar=1.0`	Cd0
induced_drag_scalar	`induced_drag_scalar=1.0`	Cdi

[weight_and_balance] section

The weight and center of gravity of the aircraft can be affected through the following parameters. The sign convention for positions is positive equals longitudinally forward, laterally to the right, and vertically upward.

Parameter	Example (from C182 aircraft.cfg)	Units / format	Description
empty_weight	`empty_weight=1810`	pounds	Total weight (in pounds) of the aircraft minus usable fuel, passengers, cargo, and expendable armament (Combat Flight Simulator aircraft). If not specified, the value previously set in the .air file will be used.
reference_datum_position	`reference_datum_position= 3.6, 0, 0`	feet,feet,feet	Offset (in feet) of the aircraft's reference datum from the standard Flight Simulator center point, which is on the centerline ¼ chord aft of the leading edge. By setting the Reference Datum Position, actual aircraft loading data can be used directly according to the aircraft's manufacturer. If not specified, the default is 0,0,0 (Flight Simulator's default center).
empty_weight_CG_position	`empty_weight_CG_position= -3.0, 0, 0`	feet,feet,feet	Offset (in feet) of the center of gravity of the basic empty aircraft (no fuel, passengers, or baggage) from the Reference Datum Position.

station_load.0	station_load.0=170, -3.0, -1.5, 0.0	pounds,feet,feet,feet	Specifies the weight and position of passengers or payload at a station specified with a unique number, station_load.N. The first parameter number on each line specifies the weight (in pounds), followed by the offset (in feet) of the station (longitudinal, lateral, and vertical) from the Reference Datum Position. The addition of stations results in a corresponding change in aircraft flight dynamics due to the change of the total weight and moments of inertia.
station_load.1	station_load.1=170, -3.0, 1.5, 0.0	pounds,feet,feet,feet	Same as above
station_load.2	station_load.2=160, -6.2, -1.5, 0.0	pounds,feet,feet,feet	Same as above
station_load.3	station_load.3=160, -6.2, 1.5, 0.0	pounds,feet,feet,feet	Same as above
station_load.4	station_load.4=102, -8.0, 0.0, 0.0	pounds,feet,feet,feet	Same as above
max_number_of_stations	max_number_of_stations=50	number	Max Number of Stations specifies the maximum number of stations Flight Simulator will calculate when the aircraft is loaded. This allows an unlimited number of stations to be specified. Note that an excessively large number here results in a longer load time for the aircraft when selected, although there is no effect on real-time performance.

Moments of Inertia

A moment of inertia (MOI) defines the mass distribution about an axis of an aircraft. A moment of inertia for a particular axis is increased as mass is increased and/or as the given mass is distributed farther from the axis. This is largely what determines the inertial characteristics of the aircraft.

The following weight and balance parameters define the MOIs of the empty aircraft, meaning that the value should not reflect fuel, passengers, baggage, or expendable armament (Combat Flight Simulator aircraft). Flight Simulator determines the total MOIs with these influences within the simulation. The units are slug - ft^2. Omission of a parameter will result in Flight Simulator defaulting to the value set in the .air file, located in the aircraft's container folder.

Parameter	Example (from C182.air file)	Description
empty_weight_pitch_MOI	`empty_weight_pitch_MOI=1400.0`	The moment of inertia (MOI) about the lateral axis.
empty_weight_roll_MOI	`empty_weight_roll_MOI=1137.0`	The moment of inertia (MOI) about the longitudinal axis.
empty_weight_yaw_MOI	`empty_weight_yaw_MOI=2360.0`	The moment of inertia (MOI) about the vertical axis.

These values can be estimated with the following formula:

MOI = EmptyWeight * (D^2 / K)

Where:

	Pitch	Roll	Yaw
D =	Length (feet)	Wingspan (feet)	0.5* (Length+Wingspan)
K =	810	1870	770

This formula yields only rough estimates. Actual values vary based on aircraft material, installed equipment, and number of engines and their positions.

[piston_engine] and [turboprop_engine] sections

Power can be scaled from the calculated values generated for piston and turboprop engines. The amount of power generated by an engine and the power required for a propeller to turn through the air determine the increase and decrease of the RPM. Power can be scaled from the calculated values generated for piston and turboprop engines. Changing the `power_scalar` value affects the amount of power delivered by the engine to the propeller shaft.

Note: This parameter is adjustable using the Flight Simulator 2000 Aircraft Editor Application.

Parameter	Example (from C182 aircraft.cfg)
power_scalar	`power_scalar = 1.0`

[propeller] and [jet_engine] sections

The thrust generated by a given propeller is a function of the power delivered through the propeller shaft, RPM, blade angle, airplane speed, and ambient density. The `thrust_scalar` parameter scales the calculated thrust for propeller and pure jet engines.

Note: This parameter is adjustable using the Flight Simulator 2000 Aircraft Editor Application.

Parameter	Example (from C182 aircraft.cfg)
thrust_scalar	`thrust_scalar = 1.0`

[ground_reaction] section

These parameters define the characteristics of the landing gear. Adjust these values, especially the center values, to fix "bouncy" tail wheel aircraft.

Static compression

Static compression is the distance, in feet, that the landing gear is compressed when the empty aircraft is at rest on the ground.

Parameter	Example (from C182 aircraft.cfg)	Units	Description
main_gear_static_compression	`main_gear_static_compression = 0.3`	feet	The distance the main gear strut(s) is/are compressed when at rest on the ground.
center_gear_static_compression	`center_gear_static_compression = 0.3`	feet	The distance the center gear strut(s) is/are compressed when at rest on the ground.

Damping ratio

The damping ratio describes the damping of ground reaction oscillations. A damping ratio of 1.0 is considered critically damp, meaning there will be little or no oscillation. A damping ratio of

0.0 is considered undamped, meaning that the oscillations will continue with approximately a constant magnitude. A negative damping ratio results in increasing magnitude of the oscillations, making ground handling unstable. Damping ratios greater than 1.0 may also cause instability due to excessive damping forces. Typical damping ratios range between 0.65 and 0.95. If the following parameters are omitted, Flight Simulator will use the values in the aircraft's .air file.

Parameter	Example (from C182 aircraft.cfg)	Description
main_gear_damping_ratio	`main_gear_damping_ratio=0.7`	The ratio of damping applied to oscillations of the main gear strut(s).
center_gear_damping_ratio	`center_gear_damping_ratio=0.7`	The ratio of damping applied to oscillations of the center gear strut(s).

[electrical] section

These parameters configure the characteristics of the aircraft's electrical system and its components. Each aircraft has a battery as well as an alternator or generator for each engine.

Below is a table of `[electrical]` section parameters shown with typical default values (the values Flight Simulator uses if the parameters are omitted). The default Min Voltage equals 0.7*Max Battery Voltage. The list of components also reflects all of the systems that are currently linked to the electrical system. If a component is included in the list but the aircraft does not actually have that system, the component is simply ignored.

Parameter	Example (with default values that the simulation uses if the lines are omitted)	Units (see note below)	Description
max_battery_voltage	`max_battery_voltage = 24.0`	volts	The maximum voltage to which the battery can be charged. It is also the voltage available from the battery when the aircraft is initialized. The battery voltage will decrease from this if the generators or alternators are not supplying enough current to meet the demand of the active components.

generator_alternator_voltage	generator_alternator_voltage = 28.0	volts	The voltage provided by a fully functioning alternator or generator.
max_generator_alternator_amps	max_generator_alternator_amps = 60.0	amps	The maximum current (amperage) provided by a fully functioning alternator or generator.
flap_motor	flap_motor = 0, 5 , 17.0	Bus type, max amp load, min voltage	See notes below.
gear_motor	gear_motor = 0, 5 , 17.0	Bus type, max amp load, min voltage	See notes below.
autopilot	autopilot = 0, 5 , 17.0	Bus type, max amp load, min voltage	See notes below.
avionics_bus	avionics_bus = 0, 10, 17.0	Bus type, max amp load, min voltage	See notes below.
avionics	avionics = 1, 5 , 17.0	Bus type, max amp load, min voltage	See notes below.
pitot_heat	pitot_heat = 0, 15 , 17.0	Bus type, max amp load, min voltage	See notes below.
additional_system	additional_system = 0, 20, 17.0	Bus type, max amp load, min voltage	See notes below.
marker_beacon	marker_beacon = 1, 2 , 17.0	Bus type, max amp load, min voltage	See notes below.
gear_warning	gear_warning = 0, 2 , 17.0	Bus type, max amp load, min voltage	See notes below.
fuel_pump	fuel_pump = 0, 5 , 17.0	Bus type, max amp load, min voltage	See notes below.
starter1	starter1 = 0, 20, 17.0	Bus type, max amp load, min voltage	See notes below.
starter2	starter2 = 0, 20, 17.0	Bus type, max amp load, min voltage	See notes below.
starter3	starter3 = 0, 20, 17.0	Bus type, max amp load, min voltage	See notes below.
starter4	starter4 = 0, 20, 17.0	Bus type, max amp load, min voltage	See notes below.

light_nav	`light_nav = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_beacon	`light_beacon = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_landing	`light_landing = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_taxi	`light_taxi = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_strobe	`light_strobe = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_panel	`light_panel = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
prop_sync	`prop_sync = 0, 15 , 17.0`	Bus type, max amp load, min voltage	See notes below.
auto_feather	`auto_feather = 0, 15 , 17.0`	Bus type, max amp load, min voltage	See notes below.
auto_brakes	`auto_brakes = 0, 15 , 17.0`	Bus type, max amp load, min voltage	See notes below.
standby_vacuum	`standby_vacuum = 0, 15 , 17.0`	Bus type, max amp load, min voltage	See notes below.
hydraulic_pump	`hydraulic_pump = 0, 2 , 17.0`	Bus type, max amp load, min voltage	See notes below.
fuel_transfer_pump	`fuel_transfer_pump = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
propeller_deice	`propeller_deice = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_recognition	`light_recognition = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_wing	`light_wing = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.
light_logo	`light_logo = 0, 5 , 17.0`	Bus type, max amp load, min voltage	See notes below.

Notes:

- **Bus Type** specifies which bus in the electrical system the component is connected to, according to the following bus type codes:

Bus Type	Bus
0	Main Bus (most components connected here)
1	Avionics Bus
2	Battery Bus
3	Hot Battery Bus (bypasses Master switch)
4	Generator/Alternator Bus 1 (function of engine 1)
5	Generator/Alternator Bus 2 (function of engine 2)
6	Generator/Alternator Bus 3 (function of engine 3)
7	Generator/Alternator Bus 4 (function of engine 4)

- **Max Amp Load** is the current required to power the component, and of course becomes the additional load on the electrical system.
- **Min Voltage** is the minimum voltage required from the specified bus for the component to function.

[pitot_static] section

The `vertical_speed_time_constant` parameter can be used to tune the lag of the Vertical Speed Indicator for the aircraft:

- Increasing the time constant decreases the lag, making the gauge react more quickly.
- Decreasing the time constant increases the lag, making the gauge react more slowly.
- A value of 0 effectively causes the indication to freeze. If an instantaneous indication is desired, use an excessively large value, such as 99.
- If the line is omitted, the default value is 2.0.

Parameter	Example (from Bell206b aircraft.cfg)	Description
vertical_speed_time_constant	`vertical_speed_time_constant = 2.0`	Increases or decreases the lag of the vertical speed indicator.

[helicopter] section

New in Flight Simulator 2000, the `low_realism_stability_scale` parameter scales the stability of the Bell 206B helicopter in low realism settings, to make the aircraft easier to fly. The stability factor is broken down into three components: pitch, roll, and yaw damping.

Here's how the simulation uses this parameter:

1. The stability factor is scaled according to the pitch, roll, and yaw values set in the `[helicopter]` section of the aircraft.cfg. For example, increasing the first value (pitch) to 1.1 increases the pitch-damping factor by 10%.

2. Then, the stability factor is scaled by the General Flight Model Realism slider in the Realism Settings dialog box. At the highest realism setting, it is scaled to 0% (no additional damping); at the minimum setting, it is scaled to 100%.

Changes to the stability factor in the .cfg have their largest effect when the Realism Setting is set to minimum, and have no effect when Realism Setting is set to maximum.

Note: Increasing these values excessively will result in excessive damping, making it hard to control the helicopter.

Parameter	Example (from Bell206b aircraft.cfg)	Format	Description
low_realism_stability_scale	`low_realism_stability_scale = 1.0, 1.0, 1.0`	pitch, roll, yaw	Scales helicopter stablility in low realism settings.

The kneeboard content files

Introduction

The text displayed on the Key Commands tab of the kneeboard is located in the main Aircraft folder, and is named KNEEBOARD_KEYS.txt

The text displayed on the Checklists, Reference, and Notes tabs of the kneeboard is aircraft-specific, and saved in text files in the aircraft containers (aircraft folders) associated with each aircraft:

- The file for the Checklists tab is named *aircraftname*_check.txt where *aircraftname* is the name of the aircraft (e.g., EXTRA300S_CHECK.txt).
- The file for the Reference tab is named *aircraftname*_ref.txt where *aircraftname* is the name of the aircraft (e.g., EXTRA300S_ref.txt).
- The file for the Notes tab is named *aircraftname*_notes.txt where *aircraftname* is the name of the aircraft (e.g., EXTRA300S_notes.txt). This file is automatically generated the first time a user clicks on the Notes tab of the kneeboard while flying that specific aircraft. Although a user can write directly into the Notes tab of the kneeboard while flying, editing the _notes.txt files directly is an easy way to "pre-populate" them with text.

Modifying the kneeboard tab files

You can open the kneeboard's .txt files with NotePad or any other text editing program that can read and save files in ASCII or text format. You can use spaces and tabs to align the text, but it will probably take some time to get the layout right. Make a change, then see how it looks when displayed on the kneeboard. If you don't like what you see (if words wrap to the next line, for instance), make changes and look again.

After you've made changes to any kneeboard .txt files, be sure to either restart Flight Simulator or select a different aircraft to reload the kneeboard. If you've modified the Extra 300S Checklists tab, for example, be sure to select another aircraft from the Aircraft menu, and then switch back to the Extra 300S in order to see the changes you've made.

About Flight Simulator 98/Combat Flight Simulator aircraft checklists

If you've imported an aircraft from Flight Simulator 98 or Combat Flight Simulator, you'll find a file named *aircraftname*_check.cfg in the aircraft's container folder. This is the checklist file in Flight Simulator 98 or Combat Flight Simulator format. Flight Simulator 2000 automatically converts the file and creates a new _CHECK.txt file that will display the aircraft's multiple checklist pages on the single Checklists tab of the kneeboard.

Creating a Checklists tab for an imported aircraft's kneeboard

After importing a Flight Simulator 98 or Combat Flight Simulator aircraft you can create a new checklist for the Flight Simulator 2000 kneeboard. Make backup copies of the original files so you can restore them if you make mistakes.

To make a Flight Simulator 2000 checklist

1. Using NotePad or another text editing program, create a .txt file containing your checklist.

2. Save the checklist with the same name as the Flight Simulator 98 or Combat Flight Simulator checklist file, but with a .txt extension instead of a .cfg extension. For example, to create a new checklist file for a P-51D imported from Combat Flight Simulator, the new file name would be P51d_CHECK.txt. Make sure you save the checklist file in the correct aircraft container (aircraft folder).

3. Open the appropriate aircraft.cfg file with a text editor. At the end of the appropriate `[fltsim.]` section (configuration group), add the line `kb_checklists=[filename]` where `[filename]` is the name of the checklist file you just created. Do not type the file's extension (.txt). For example, the correct line of the aircraft.cfg file for the P-51D would read `KB_checklists=P51d_CHECK`

4. Either restart Flight Simulator or select a different aircraft to reload the kneeboard. If you've modified the imported P-51D's Checklists tab, for example, be sure to select another aircraft from the Aircraft menu, and then switch back to the P-51D in order to see the changes you've made.

You can use spaces and tabs to align the text, but it will probably take some time to get the layout right. Make a change, then see how it looks when displayed on the kneeboard. If you don't like what you see (if words wrap to the next line, for instance), make changes and look again.

The panel.cfg file

The panel.cfg file is located in an aircraft's Panel folder, and defines the characteristics of the aircraft's cockpit, including window settings, view settings, and gauges. For a full explanation of the structure of a panel.cfg file and instructions for editing it, look for the Panels SDK.

The model.cfg file

Introduction

The model.cfg file is located in an aircraft's Model folder, and specifies which visual models (.mdl files) to render during normal flight and during a crash, as well as the visual model color scheme. A model.cfg file has `[models]` and `[colors]` parameters. You should look at actual model.cfg files to become more familiar with the structure and syntax of them. They can be found in the individual aircraft containers, and viewed using any text editor.

[models] section

These parameters specify which visual model to render during normal flight, and during a crash.

Parameter	Example (from C182S model.cfg)	Description
normal	`normal=Cessna182S_n`	Determines which visual model (.mdl file) to render during normal flight.
crash	`crash=Cessna182S_c`	Determines which visual model (.mdl file) to render during a crash.

[colors] section

Colors parameters are not used for Flight Simulator 2000 aircraft.

For imported Flight Simulator 98 and Combat Flight Simulator aircraft, these parameters define the *untextured* colors of distinct aircraft parts. The aircraft textures are displayed on top of these colored parts. Color parameters are numbered sequentially beginning with 00. Each numbered parameter refers to a different part of the aircraft. The designer may or may not have described the part in commented text before the parameter line in the code.

An example (taken from the Combat Flight Simulator P51d model.cfg file):

```
[colors]

; body - predominate color on plane

00=WHITE
```

; wing - color of wing, flaps, ailerons

01=WHITE

; altbody - two tone body part

02=WHITE

Note: Some model designers choose to have the model ignore the colors specified in the model.cfg. Thus, depending on how the designer created the model, some of the following colors may not be valid.

In general, you can alter (either completely or in part) the color scheme of an aircraft by using the following colors:

- BLACKBLUE
- BRICKBRIGHT_AQUA
- BRIGHT_BLUEBRIGHT_DKGRAY
- BRIGHT_GRAYBRIGHT_GREEN
- BRIGHT_LTGRAYBRIGHT_ORANGE
- BRIGHT_REDBRIGHT_WHITE
- BRIGHT_YELLOWBROWN
- DARK_BLUEDARK_BRICK
- DARK_BROWNDARK_GREEN
- DARK_OLIVEDARK_ORANGE
- DARK_REDDARK_TAN
- DARK_YELLOWDKGRAY
- GRAYGREEN
- LIGHT_BLUELIGHT_BRICK
- LIGHT_BROWNLIGHT_GREEN
- LIGHT_OLIVELIGHT_ORANGE
- LIGHT_REDLIGHT_TAN
- LIGHT_YELLOWLTGRAY
- MED_BLUEMED_BRICK
- MED_BROWNMED_GREEN
- MED_OLIVEMED_ORANGE
- MED_REDMED_TAN
- MED_YELLOWOLIVE
- ORANGERED
- TANWHITE
- YELLOW

The sound.cfg file

Introduction

The sound.cfg file is located in an aircraft's Sound folder, and defines the sounds to use for that aircraft (such as the sound of the engine at various RPMs, the sound of the landing gear going down, and so on). This file also specifies attributes for each sound that determine exactly how the sound is played.

Many aircraft sounds in Flight Simulator are composed of multiple .wav files (called a "sound list") that are linked to one another, processed in sequence, and then played as a group. These sounds are updated by the simulator's sound engine every time the Flight Simulator screen refreshes (once every frame).

As you read this section, you should look at actual sound.cfg files to become more familiar with the structure and syntax of them. They can be found in the Sound subfolders of the aircraft containers, and viewed using any text editor.

To hear any particular component of a sound in its "pure" form (unaffected by the attributes in the sound.cfg file), just play the .wav file referenced in the sound.cfg (.wav files are located in either the Sound folder in an aircraft's container, or in the FS2000\Sound folder).

Notes:

- For clarity, the naming convention used for the default Flight Simulator 2000 aircraft engine sounds described in the following sections was to put an "x" at the beginning of all external sound (heard in Spot and Tower views) headers, and to number consecutive sound headers (e.g., `[SHUTDOWN]`, `[SHUTDOWN.1]`). The specific header names used does not matter, as long as they are consistent in the `[SOUND_ENGINE]` section and across link parameters. Additionally, the order of the sections within the sound.cfg file does not matter, nor does the order of the parameters within a section.

- Flight Simulator 2000 can load most PCM .wav formats, as well as compressed formats. When a compressed format is used, Flight Simulator uses Audio Compression Manager (ACM), an audio compression module included in Windows 95 and 98, to load the file. If you include compressed files, make sure that they are in a format supported (by default) by Windows 95 and 98, such as ADPCM. To determine the compression formats supported by your system:

    1. Click the **Start** button, point to **Settings**, then click **Control Panel.**
    2. Double-click the **Multimedia** icon.
    3. Click the **Devices** tab.
    4. Double-click the **Audio Compression Codecs** header to expand the list of supported formats.

Following is a description of each section of a sound.cfg file.

[fltsim] section

This parameter distinguishes Flight Simulator 2000 sound.cfg files from previous version files. All sound.cfg files created for Flight Simulator 2000 should have this section, and a parameter that reads:

```
product_code=FSIM
```

Note: The Bell 206B helicopter included with Flight Simulator 2000 does not have startup or shutdown sounds. Thus, as far as sound is concerned, it is modeled like a Flight Simulator 98 aircraft, and its sound.cfg file has no `[FLTSIM]` section. Note too that for this aircraft, instead of multiple engine-related sounds, there are only `[JET_ENGINE]` and `[ENGINE_EXT]` sounds (for the sound of the engine from the cockpit and from outside, respectively).

[sound_engine] section

These parameters concern the engine sounds of an aircraft. They specify the number of engines the aircraft has, and the sound lists the simulation should use to create the aircraft's engine sounds. Each sound list is referenced by the header of the first sound in the list (additional sounds are linked to in sequence from that first sound). The individual sounds in a list are defined in their own sections within the sound.cfg file (see the *Specific engine sound parameters (sound lists)* section below).

[sound_engine] parameters for jet aircraft

The following table describes the parameters used to define the sounds of a jet engine in the `[sound_engine]` section of a sound.cfg file:

Parameter	Example	Description
number_of_engines	`number_of_engines=2`	How many engines the aircraft has.
eng1_combustion	`eng1_combustion=COMBUSTION.1.00`	Points to the first sound in a sound list of engine 1 combustion sounds.
eng2_combustion	`eng2_combustion=COMBUSTION.2.00`	Points to the first sound in a sound list of engine 2 combustion sounds.
eng1_jet_whine	`eng1_jet_whine=JET_WHINE.1.00`	Points to the first sound in a sound list of engine 1 jet whine sounds.
eng2_jet_whine	`eng2_jet_whine=JET_WHINE.2.00`	Points to the first sound in a sound list of engine 2 jet whine sounds.
eng1_starter	`eng1_starter=starterA`	Points to the first sound in a sound list of engine 1 starter sounds.

eng2_starter	`eng2_starter=starterB`	Points to the first sound in a sound list of engine 2 starter sounds.
eng1_shutdown	`eng1_shutdown=shutdownA`	Points to the first sound in a sound list of engine 1 shutdown sounds.
eng2_shutdown	`eng2_shutdown=shutdownB`	Points to the first sound in a sound list of engine 2 shutdown sounds.
eng1_combustion_start	`eng1_combustion_start=combstartA`	Points to the first sound in a sound list of engine 1 combustion start sounds.
eng2_combustion_start	`eng2_combustion_start=combstartB`	Points to the first sound in a sound list of engine 2 combustion start sounds.
eng1_non_combustion	`eng1_non_combustion=NON_COMBUSTION.1.00`	Points to the first sound in a sound list of engine 1 non-combustion sounds (the isolated sounds of the engine's moving parts).
eng2_non_combustion	`eng1_non_combustion=NON_COMBUSTION.1.00`	Points to the first sound in a sound list of engine 2 non-combustion sounds (the isolated sounds of the engine's moving parts).

[sound_engine] parameters for turboprop aircraft

The following table describes the parameters used to define the sounds of a turboprop engine in the `[sound_engine]` section of a sound.cfg file:

Parameter	Example	Description
number_of_engines	`number_of_engines=2`	How many engines the aircraft has.
eng1_combustion	`eng1_combustion=COMBUSTION.1.00`	Points to the first sound in a sound list of engine 1 combustion sounds.
eng2_combustion	`eng2_combustion=COMBUSTION.2.00`	Points to the first sound in a sound list of engine 2 combustion sounds.
eng1_jet_whine	`eng1_jet_whine=JET_WHINE.1.00`	Points to the first sound in a sound list of engine 1 jet whine sounds.
eng2_jet_whine	`eng2_jet_whine=JET_WHINE.2.00`	Points to the first sound in a sound list of engine 2 jet whine sounds.
eng1_prop	`eng1_prop=PROP.1.00`	Points to the first sound in a sound list of engine 1 prop sounds.

eng2_prop	`eng2_prop=PROP.2.00`	Points to the first sound in a sound list of engine 2 prop sounds.
eng1_starter	`eng1_starter=starterA`	Points to the first sound in a sound list of engine 1 starter sounds.
eng2_starter	`eng2_starter=starterB`	Points to the first sound in a sound list of engine 2 starter sounds.
eng1_shutdown	`eng1_shutdown=shutdownA`	Points to the first sound in a sound list of engine 1 shutdown sounds.
eng2_shutdown	`eng2_shutdown=shutdownB`	Points to the first sound in a sound list of engine 2 shutdown sounds.
eng1_combustion_start	`eng1_combustion_start=combstartA`	Points to the first sound in a sound list of engine 1 combustion start sounds.
eng2_combustion_start	`eng2_combustion_start=combstartB`	Points to the first sound in a sound list of engine 2 combustion start sounds.
eng1_non_combustion	`eng1_non_combustion=NON_COMBUSTION.1.00`	Points to the first sound in a sound list of engine 1 non-combustion sounds (the isolated sounds of the engine's moving parts).
eng2_non_combustion	`eng1_non_combustion=NON_COMBUSTION.1.00`	Points to the first sound in a sound list of engine 2 non-combustion sounds (the isolated sounds of the engine's moving parts).

[sound_engine] parameters for piston aircraft

The following table describes the parameters used to define the sounds of a piston engine in the `[sound_engine]` section of a sound.cfg file:

Parameter	Example	Description
number_of_engines	`number_of_engines=1`	How many engines the aircraft has.
eng1_combustion	`eng1_combustion=COMBUSTION.1.00`	Points to the first sound in a sound list of engine 1 combustion sounds.
eng1_starter	`eng1_starter=starter`	Points to the first sound in a sound list of engine 1 starter sounds.
eng1_combustion_start	`eng1_combustion_start=combstart`	Points to the first sound in a sound list of engine 1 combustion start sounds.

eng1_shutdown	`eng1_shutdown=shutdown`	Points to the first sound in a sound list of engine 1 shutdown sounds.
eng1_prop	`eng1_prop=PROP.1.00`	Points to the first sound in a sound list of engine 1 prop sounds.
eng1_non_combustion	`eng1_non_combustion=NON_COMBUSTION.1.00`	Points to the first sound in a sound list of engine 1 non-combustion sounds (the isolated sounds of the engine's moving parts).

Specific engine sound sections (sound lists)

The parameters in these sections (e.g., `[STARTER]`, `[XSTARTER]`, `[SHUTDOWN]`, `[SHUTDOWN.1]`, `[COMBSTART]`, etc.) are used to define the specific aircraft engine sounds referenced by the parameters in the `[SOUND_ENGINE]` section. Each sound has its own section in the sound.cfg with a bracketed header, and is linked to the next sound in the sound list with the `link=` parameter.

The following table describes parameters used in the engine sound sections of an aircraft's sound.cfg file:

Parameter	Example	Description
filename	`filename=cela`	The name of the .wav file to play. The .wav extension should not be specified. Flight Simulator 2000 searches the Sound folder in the specific aircraft container first, and then (if the file isn't found) searches the Flight Simulator 2000 Sound folder.
flags	`flags=0`	Flags have different functions when associated with different sounds. **For all sounds** 0 = no flag 1 = disable sound **For `[Combustion]` sounds:** 2 = damaged 4 = boost (Imported Combat Flight Simulator aircraft only) 8 = jet engine rumble sound **For `[Prop]` sounds** 2 = max prop pitch 4 = min prop pitch 8 = min reverse prop pitch

viewpoint	`viewpoint=1`	Determines whether the sound is audible in internal cockpit views (specified by the value 1), or in external spot and tower views (specified by the value 2).
vparams	`vparams=0.000000,53.600000,0.174000,55.200000, 0.289000,12.000000,0.530000,0.000000,0.530000, 0.000000,0.530000,0.000000,0.530000,0.000000`	Defines the sound's amplitude envelope. Represents the sound's volume as a function. Each pair of values specified in vparams represents a single point; you can use up to 8 points to describe the amplitude envelope. The first number in the pair is a generic value that can range from 0.0 to 1.0, the second number specifies volume. (The units for volume are linear, with a value of 50 meaning –3dB of attenuation, and 0 meaning silence.)
rparams	`rparams=0.000000,1.000000,0.264000,1.110000`	Defines the sound's pitch envelope. Represents the sound's relative pitch (and, invariably, the playback speed) as a function of a generic value that can range from 0.0 to 1.0. Each pair of values specified in rparams represents a single point; you can use up to 2 points to describe the pitch envelope. The format and behavior of rparams is similar to vparams, except that the second value of each point represents a pitch scaler. A value of 1.0 specifies that the sound file is played at unity pitch. A value of 2.0 specifies that the file is played an octave higher and twice as fast.
panning	`panning=0`	Determines where the sound is placed in the stereo field: 0 = center -10,000 = full left 10,000 = full right
link	`link=COMBUSTION.1.01`	References the next sound in a sound list (by section heading name). Most engine sounds are made up of several .wav files, and each .wav file has its own section in the .cfg file. The last sound in a sound list has no link parameter. The order of sounds in a list is not important.

[wind_sound] section

Wind sounds are used to add realism to the sounds of aircraft. Wind sound is also the predominant sound used for sailplanes. Each wind sound can be volume- and pitch-modulated with airspeed.

The following table describes the parameters used in the wind section of an aircraft's sound.cfg file:

Parameter	Example (from Schweizer sound.cfg)	Description
filename	`filename=snwind5`	The name of the .wav file to play. The .wav extension should not be specified. Flight Simulator 2000 searches the Sound folder in the specific aircraft container first, and then (if the file isn't found) searches the Flight Simulator 2000 Sound folder.
flags	`flags=0`	Flags have different functions when associated with different sounds. **For all sounds** 0 = no flag 1 = disable sound There are no wind sound-specific flags.
minimum_speed	`minimum_speed=0.1`	The minimum speed (in KTAS) used by the `_volume` and `_rate` parameters.
maximum_speed	`maximum_speed=160.0`	The speed (in KTAS) above which the sound has constant volume and pitch. Specified in KTAS units.
minimum_volume	`minimum_volume=7000`	The lowest possible volume--if the sound drops below the minimum volume specified, it will not be heard. If the aircraft's speed is between `minimum_speed` and `maximum_speed`, playback volume is interpolated. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)
maximum_volume	`maximum_volume=10000`	Specifies the highest possible volume--the sound never exceeds the volume specified. If the aircraft's speed is between `minimum_speed` and `maximum_speed`, playback volume is interpolated. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)
minimum_rate	`minimum_rate=0.50`	Specifies the minimum rate at which the sound is played. If the aircraft's speed is between `minimum_speed` and `maximum_speed`, the playback rate is interpolated between the `minimum_rate` and `maximum_rate` values.

maximum_rate	`maximum_rate=1.25`	Specifies the maximum rate at which the sound is played. If the aircraft's speed is between `minimum_speed` and `maximum_speed`, the playback rate is interpolated between the `minimum_rate` and `maximum_rate` values.

Ground sound sections

Ground sounds include:

```
[CENTER_TOUCHDOWN]
[AUX_TOUCHDOWN]
[LEFT_TOUCHDOWN]
[RIGHT_TOUCHDOWN]
[FUSELAGE_SCRAPE]
[LEFT_WING_SCRAPE]
[RIGHT_WING_SCRAPE]
[AUX1_SCRAPE]
[AUX2_SCRAPE]
[XTAIL_SCRAPE]
[GROUND_ROLL]
```

Some ground sounds consist of multiple sets of .wav files (sound lists), and each set corresponds to a unique combination of surface types. Each ground sound can be volume- and pitch-modulated with airspeed.

The following table describes the additional parameters used in the ground sound sections of an aircraft's sound.cfg file:

Parameter	Example (from Extra300 sound.cfg)	Description
filename	`filename=cnroll2`	Specifies the name of the .wav file to play. The .wav extension should not be specified. Flight Simulator 2000 searches the Sound folder in the specific aircraft container first, and then (if the file isn't found) searches the Flight Simulator 2000 Sound folder. **Note:** If ground sound filename parameters have comma separated filenames (e.g. `filename=cmtouch1, cmtouch2, cmtouch3`), the simulator code will randomly choose to play one of the listed .wav files.

flags	`flags=125218`	Flags have different functions when associated with different sounds. **For all sounds** 0 = no flag 1 = disable sound **For ground sounds** By flagging a sound for a particular ground surface type, you tell the simulation to play that sound when the aircraft comes into contact with that surfaces type or types. Ground sound flags include: 2 = concrete 4 = soft, bumpy ground (landable) 8 = water 16 = very bumpy grass & mud (crashable) 32 = asphalt 64 = short grass 128 = long grass 256 = hard turf 512 = snow 1024 = ice 2048 = urban 4096 = forest 8192 = dirt runway 16384 = coral runway 32768 = gravel runway 65536 = oil treated (tar&chip) runway 131072 = steel mats (steel mesh) temporary runway Note that these values are powers of 2 so that they represent bits. For instance, the `[GROUND_ROLL]` section of the 182S sound.cfg file has the line `flags=125218` This is 11110100100100010 in binary, and maps to concrete+asphalt+hard turf, etc.
minimum_speed	`minimum_speed=3`	Specifies the minimum speed (in KTAS) used by the `_volume` and `_rate` parameters.
maximum_speed	`maximum_speed=55`	Specifies the speed (in KTAS) above which the sound has constant volume and pitch. Specified in KTAS units.
minimum_volume	`minimum_volume=6000`	Specifies the lowest possible volume--if the sound drops below the minimum volume specified, it will not be heard. If the aircraft's speed is between `minimum_speed` and `maximum_speed`, playback volume is interpolated. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)

maximum_volume	maximum_volume=10000	Specifies the highest possible volume--the sound never exceeds the volume specified. If the aircraft's speed is between minimum_speed and maximum_speed, playback volume is interpolated. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)
minimum_rate	minimum_rate=0.80	Specifies the minimum rate at which the sound is played. If the aircraft's speed is between minimum_speed and maximum_speed, the playback rate is interpolated between the minimum_rate and maximum_rate values.
maximum_rate	maximum_rate=1.60	Specifies the maximum rate at which the sound is played. If the aircraft's speed is between minimum_speed and maximum_speed, the playback rate is interpolated between the minimum_rate and maximum_rate values.
link	link=GROUND_ROLL2	References the next sound in a sound list (by section heading name). Some ground sounds are made up of several .wav files, and each .wav file has its own section in the .cfg file.

Other sound sections

Other sounds used by Flight Simulator aircraft include:

[GEAR_UP_WARNING_SOUND]

[STALL_WARNING]

[OVERSPEED_WARNING_SOUND]

[GLIDESLOPE_WARNING_SOUND]

[AP_DISENGAGE_SOUND]

[GEAR_DOWN]

[GEAR_UP]

[FLAPS]

[CRASH_SOUND]

[SPLASH_SOUND]

The following table describes the parameters used in other sound sections of an aircraft's sound.cfg file:

Parameter	Example (from Extra300 sound.cfg)	Description
filename	`filename=snwind5`	Specifies the name of the .wav file to play. The .wav extension should not be specified. Flight Simulator 2000 searches the Sound folder in the specific aircraft container first, and then (if the file isn't found) searches the Flight Simulator 2000 Sound folder. **Note:** `[CRASH_SOUND]` and `[SPLASH_SOUND]` `filename` parameters have comma separated filenames (e.g. `filename=encrash1,encrash2`). In these instances, the simulator code will randomly choose to play one of the listed .wav files.
flags	`flags=0`	Flags have different functions when associated with different sounds. **For all sounds** 0 = no flag 1 = disable sound
initial_volume	`initial_volume=9000`	Specifies the volume at which the sound starts. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)
minimum_volume	`minimum_volume=7800`	Specifies the lowest possible volume--if the sound drops below the minimum volume specified, it will not be heard. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)
maximum_volume	`maximum_volume=9000`	Specifies the highest possible volume--the sound never exceeds the volume specified. (Volume is specified in 1/100dB units, with a value of 10,000 being the maximum possible volume.)

The Texture folder

An aircraft's textures are defined by the .bmp files in the aircraft's Texture folder, and are "projected" onto the aircraft's parts as specified in the aircraft's visual model (.mdl) file, located in the Model folder. Texture file names must correspond to the texture files that are referenced in the .mdl file. If the file names don't correspond, the textures won't be visible.

Flight Simulator 2000 texture files are "mipmapped". A mipmapped texture consists of a sequence of images, each of which is a progressively lower resolution, prefiltered representation of the same image. Mipmapping is a computationally low-cost way of improving the quality of rendered textures. Each prefiltered image, or level, in the mipmap is a power of two smaller than the previous level. A high-resolution level is used for objects that are close to the viewer. Lower-resolution levels are used as the object moves farther away.

To edit a mipmapped texture, you'll need to use *Image Tool*, an image editing application included with the Flight Simulator 2000 Terrain SDK. Be sure to save a copy of the original file before attempting to modify it.

A texture can also be edited using a simple graphics application such as Microsoft Paint, though it will be saved as a standard .bmp file instead of a mipmapped .bmp. Flight Simulator will automatically generate the mipmaps for the texture, although these mipmaps will not be of as high a quality as mipmaps created using Image Tool.

Using aliasing

Aliasing allows multiple aircraft containers to use the same files (panels, flight models, sounds etc.). This saves disk space and makes file organization more efficient. You can alias an aircraft's panel.cfg, model.cfg, and sound.cfg files from any other aircraft's. Whereas configuration sets allow aircraft *within* a single aircraft container to share components, aliasing allows aircraft in *different* aircraft containers to share components.

To alias an aircraft's panel.cfg, model.cfg, or sound.cfg file from another aircraft's, simply change the aliasing .cfg file to read:

```
[fltsim]
alias=aircraftname\panel
```

or

```
[fltsim]
alias=aircraftname\model
```

or

```
[fltsim]
alias=aircraftname\sound
```

Flight Simulator searches for aliased files in the following order:

1. Relative path from the Aircraft folder
2. Relative path from the Flight Simulator 2000 folder

An example

Let's say you've imported a Boeing 757 aircraft from the Web into Flight Simulator 2000, but want to use the 737-400 panel when flying it. Instead of duplicating all the 737-400 panel files (Panel.cfg and .bmps) and putting them in the new 757 aircraft container, you can *alias* to them in their existing location from the 757 panel.cfg file. Just change the 757 panel.cfg file to read:

```
[fltsim]
alias=panels\B737_400\panel
```

The 757 aircraft would then use the 737-400 panel.cfg file (and hence the associated .bmps).

The syntax for aliasing model.cfg and sound.cfg files is identical.